DiffPerf: Towards Performance Differentiation and Optimization with SDN Implementation

Walid Aljoby[†], Xin Wang[†], Dinil Mon Divakaran*, Tom Z. J. Fu[§], Richard T. B. Ma[†]

[†] School of Computing, National University of Singapore

* Trustwave, Singapore

[§] Bigo Technology Pte Ltd, Singapore

{algobi, xin.wang, tbma}@comp.nus.edu.sg, dinil.divakaran@trustwave.com, fuzhengjia@gmail.com

Abstract— Continuing the current trend, Internet traffic is expected to grow significantly over the coming years, with video traffic consuming the biggest share. On the one hand, this growth poses challenges to access providers, who have to upgrade their infrastructure to meet the growing traffic demands as well as find new ways to monetize their network resources. On the other hand, despite numerous optimizations of the transport congestion control, and the switch buffer sizing and management algorithms; however, the complex interaction among all of them still leads to uncertain user performance and thus degrades user-perceived quality, under various network and traffic conditions. The culprit is the difficulty to dynamically control the amount of bandwidth allocated to each of the competing flows under bottleneck due to the algorithms lack of visibility of buffer content where the flows reside.

We address both concerns by proposing **DiffPerf**, an in-network bandwidth allocation system. At a macroscopic level, **DiffPerf** elastically allocates bandwidth to performance-centric service classes pre-defined by access provider, and at a microscopic level it relies on a lightweight learning algorithm to statistically differentiate and isolate traffic flows in each class to help them achieve better performance in an online and dynamic manner. We built two SDN-based prototypes of **DiffPerf**; one on OpenDaylight with OpenFlow Brocade hardware switches and the other with data plane Intel Tofino hardware switches. We evaluate it from an application perspective for ABR video streaming as it accounts for a majority of the Internet traffic. Our evaluations demonstrate the practicality and flexibility that **DiffPerf** provides access providers with capabilities through which a spectrum of qualities are provisioned at multiple classes and assists users within the same class in achieving better fairness and improving overall user-perceived quality.

I. INTRODUCTION

During the COVID-19 pandemic, Internet connectivity has emerged as a lifeline for society to stay connected. It enables access to a multitude of life-enhancing services and underpins new services in response to the pandemic. This shift to online services is evidenced by a sharp growth in content traffic; mobile network traffic grew 50% on average in the 12 months to September 2020 [1]. Furthermore, the emerging Internet of Things (IoT) and machine-type-communications impact significantly infrastructure’s development of access networks.

Because thriving such a range of new and diverse services is becoming more integral to everyday life, the expectations towards immersive quality of experience (QoE) are also increasing. Therefore, this traffic growth and demands for Internet services pose new challenges to content providers

(CPs) as well as Internet access providers (APs), yet introduce promising business opportunities for both of them [1].

To accommodate high traffic, CPs have been deploying wide-area infrastructures to bring content closer to users to enhance the users’ QoE, e.g., Netflix uses third-party content delivery networks (CDNs) such as Akamai and Limelight, and builds its own [2]. However, as end-users rely on Internet service providers that are known as last-mile APs for accessing the Internet, an AP’s bandwidth capacity still limits user throughput due to network congestion [3]. As for APs to sustain traffic growth, APs need to upgrade network infrastructures and expand capacities; however, their incentives depend on the business model and the corresponding mechanism used to monetize bottleneck bandwidth, which is crucial to the viability of the current Internet model in the future. A general approach used by APs is to differentiate services and prices, e.g., APs provide premium peering [4] options for CPs to choose and multiple data plans for end-users with different data usage to choose. However, the former can only be implemented with large CPs via peering agreements, while the latter does not guarantee the performance of end-users in any sense. Consequently, there exists a fundamental mismatch between the differentiated services and the underlying resource allocation that differentiates for predictable performance.

Moreover, emergence of the current 5G system and its planned successor 6G serves as enablers for APs to deploy multi-access edge computing (MEC) services for enterprise users, industrial IoT, and many more [5]. This is particularly enabled via the means of network function virtualization (NFV) and software-defined networking (SDN) with the concept of end-to-end network slicing. However, in order to realize the benefits from MEC-based services, the diversity of requirements associated with various types of use cases needs a customized and performance-centric resource allocation. Inline with this vision and to provide an elastic and automated performance-centric resource allocation, we consider an in-network class-based differentiation approach, under which network operator, e.g., Internet AP or slice provider, allocates underlying bottleneck bandwidth resources in a principled and practical manner. In this approach, the AP can enable CPs and users to choose a *service class* (SC) to join. The slice provider can also enable slices that have different bandwidth demands, e.g., based on their attached IoT devices as well as the type of IoT application, to join the appropriate SC.

Nevertheless, the traffic flows in the same service class might not perceive a fair quality due to the consequences of the complex interaction between the transport protocol and the underlying network, as shown in our experimental explorations and known by conventional wisdom [6]. Numerous optimizations of the data transport congestion control (CC), and the switch buffer sizing and active queue management (AQM) algorithms have been evolved to improve the performance, such as achieving good utilization of high-speed links with reasonable queuing delay. While extremely useful, they fall short in optimizing the performance as perceived by the end-user. Based on our experimental observations for three different types of buffer sizes (i.e., shallow, moderate, and deep), for different system parameters like the bottleneck capacity, the flow round-trip times (RTT), the number of connections, and CC algorithms, we argue that performance problem is beyond buffer sizing itself, or the type of CC algorithm, but rather highly dependent on the characteristics of traffic flows that share the buffer, or simply the buffer content. While CC algorithms, buffer sizing, and many AQM algorithms lack the visibility of buffer content that makes them prone to punishing certain flows and benefiting others, the approaches such as pFabric [7], PIAS [8], and FAB [9] enable flow-aware scheduling. They prioritize short flows over long flows in the data center network by allocating them a different amount of buffer space which assists in the control of the bandwidth utilization following application-level performance requirements. However, they remain hard to adapt to the flows in the Internet context due to its much more heterogeneous traffic patterns than in the data center.

Thus, we approach the performance issue resulted from the lack of awareness of buffer content on the bottleneck corresponding to each SC. Because video streams, the most popular service of mobile networks [10], [11], dominates the majority of content traffic in that it is predicted to make up 82% of the total downstream Internet traffic by 2022, according to Cisco Annual Internet Report [10]. Hence we tackle the impact on the user-perceived QoE of adaptive bitrate (ABR) streaming which is now the de facto standard for delivering videos on the Internet. Our approach, however, can be employed for other Internet applications.

In the ABR streaming, the video players employ ABR algorithms to adjust the video playback, shown to the user, to the network available bandwidth. In the presence of congestion, the player requests a lower bitrate to avoid freezes (i.e., rebuffering) in the middle of the video playback, which is a crucial factor in the QoE perceived by the user. However, without being aware of how competing flows (i.e., users or video clients) respond to various network states, it is difficult to control the amount of bandwidth allocated among the flows, and is thus highly likely that some video players cannot even receive the lowest requested bitrate due to packet losses caused by other aggressive flows. This in large part will hurt the user-perceived QoE since the rebuffering events are strongly correlated with losses during packet transmission. We quantify and mitigate the impact of the main parameters causing the

uncertainty of performance, throughout extensive real-world video streaming experiments. We particularly demonstrate the extent to which the interplay of flows RTT, switch buffer size, and popular CC algorithms affect user-perceived QoE.

Towards performance optimization, an approach to mitigate the performance issue caused by the invisibility of current approaches to buffer contents is to optimize the user-perceived QoE through a direct participatory approach between network and application in the Internet context. Irrespective of the number of research proposals in the past [12], [13], [14], we nevertheless still do not have any active feedback system between AP vantage points and the user. Addressing this shortcoming, we proposed DiffPerf [15], an in-network approach that appropriately employs network metrics to allocate more bandwidth for flows whose performance are highly affected by other competing flows. However, the principled bandwidth allocation that should help improve the performance of user traffic flows should proactively reflect a service performance requirements in the first place and then reactively optimizes the affected flows due to invisibility to shared buffer.

In this paper, we develop and implement a complete *in-network* system approach which extends DiffPerf bandwidth allocation model to allow expressing different preferences over various performance-centric service classes. Instead of only managing the bandwidth for a certain traffic class based on the limited model in [15] the extended model gives network operator a high flexibility on managing the bandwidth of his traffic profile at multiple levels. The extended model encompasses user traffic flows of different bandwidth demands as well as the need to per-flow performance improvement.

From a macroscopic perspective, DiffPerf dynamically allocates bandwidth to each SC according to the changing number of active flows in each SC, by maximizing the weighted α -fair utilities, which enables APs to trade-off fairness. At a microscopic level, DiffPerf uses a lightweight learning algorithm to execute a new performance-aware mechanism, called (β, γ) -fairness to further make more fine-grained bandwidth allocation, to achieve fairer performance for flows and improve overall QoE perceived by the user, as well as utilizing the aggregate capacity more efficiently.

Our main contributions are summarized as follows:

1. We derive the closed-form bandwidth allocation solution and show that this solution achieves guaranteed performance differentiation in terms of controllable ratios of the average per-flow throughput across the different SCs.
2. Within each SC, we present a neat statistical learning algorithm to differentiate and isolate flows automatically, to mitigate the bias brought by the complex interplay of TCP protocol with network latency and buffer size.
3. We derive a closed-form bandwidth allocation solution that dynamically implements the (β, γ) -fairness among the competing flows, following the output of Step 2.
4. We implement¹ two prototypes of DiffPerf. (i) By

¹Our source code is available at: <https://github.com/Aljoby/DiffPerf>

leveraging SDN capabilities, we develop a native OpenDaylight (ODL) control plane application that dynamically manages network resources, including tracking flows, inquiring flow statistics, and allocating bandwidth capacity in the OpenFlow-enabled network. (ii) Furthermore, to measure the impact of switch buffer sizes, we implement another prototype of DiffPerf on programmable Tofino switch which allows flexible buffer sizing and enables more fine-grained line-rate telemetry.

5. We carry out comprehensive evaluations of DiffPerf on real-world video transmission based on the DASH system as a mainstream of Internet video traffic.

We believe that DiffPerf offers a new avenue for APs to differentiate and optimize the performance of their users (i.e., video flows and corresponding user-perceived QoE in our current prototypes), so as to better monetize their bottleneck network resources. This will further incentivize APs to deploy more bandwidth capacity to accommodate the growth of Internet content traffic.

II. BACKGROUND AND MOTIVATION

Several approaches for sharing the bandwidth between the user traffic flows have been proposed. A simple option is bandwidth max-min flow allocation [16]. This option attempts to allocate available bandwidth equally amongst the flows, with no regard to differences in application [17], assuming that benefits amongst all applications are the same. Another option is to maximize the overall system utility [18], which at the expense of achieving a higher overall system utility, some applications may receive extremely low bandwidth. A third option is to define utility function for each flow which expresses application-layer performance measure [19]. This option attempts to allocate available bandwidth amongst the flows, based on their utilities. In other words, it tries to maximize the minimum application's performance by maximizing the minimum application's utility. However, it is generally not a simple task to obtain a utility function of an application and send it back to the network to perform utility-based flow allocation. Thus, we propose a weighted max-min fairness between a set of service classes to allow spectrum of allocations among them based on the need of service operator, as in Figure 1. Each service class can be designated to a set of users or applications of similar characteristics. Our approach can flexibly support all previous options and more by tuning certain simple parameters.

In the same service class, however, the users in the same service class do not perceive a predictable performance. That is due to the bandwidth allocation for the applications that is typically embodied at the end host using TCP, and hence it does not have global visibility about content traffic on the same buffer that precedes the bottleneck capacity corresponding to each service class. Therefore, the lack of guaranteed performance in the current Internet has fostered the development adaptive applications that adapt their bandwidth usage based on feedback from the network. Dynamic Adaptive Streaming over HTTP (DASH) is the most widely used application relies

on network throughput to download video segments and is currently used by most video content providers and CDNs.

DASH encodes a video content into different levels of bitrate and delegates the selection logic of the bitrate to the client. Depending on the estimation of the bandwidth, a DASH client reacts by switching between different bitrate levels. This DASH dynamics usually causes a throughput oscillations in response to estimation of the available network bandwidth. Thus, it affects user-perceived QoE and thus does not fully solve the network performance problem.

Figure 1: DiffPerf Framework.

Given X_s amount of bandwidth capacity allocated to n_s flows in a transport slice or service class (SC) s over a bottleneck link. We denote the set of active flows in the SC by $\mathcal{F}_s$ and the cardinality of $\mathcal{F}_s$, i.e., the number of flows in the class, by n_s . Each flow $f \in \mathcal{F}_s$ is expected to achieve an average throughput of X_s/n_s . However, the actual throughput achieved, denoted by x_f , might be significantly less than the mean. This can adversely affect the QoE that the corresponding user perceives, particularly if x_f is less than the lowest bitrate requested by the video player in case of ABR video streaming. At the last-mile bottleneck, parameters such as RTT, TCP congestion control algorithm (e.g., CUBIC v/s BBR), and buffer size affect the performance of flows [20], [21], [22]. The heterogeneity of RTTs experienced by the flows as well as the interaction of the TCP-based congestion control mechanisms that respond to the RTTs and network buffer size differently, lead to multiple competing flows achieving different throughput. This results in a different impact on the user-perceived QoE.

In Figure 2 assume all flows belong to the same service class, we present the performance of 100 competing DASH-based video streaming clients (i.e., flows), where 30% of the flows experience relatively longer RTTs than the rest. All flows run TCP BBR and share a bottleneck link connecting to a DASH server. The bottleneck link capacity is set to 120Mbps which is preceded by a switch buffer size of 1MB. Figure 2 shows that flows, on the right side of the figure, experience poor QoE under the default settings.

However, by “isolating” flows that perceived dissimilar

Figure 2: The QoE of DASH-based video streaming clients using TCP BBR with/without “isolation”.

achieved throughput, we observed that DASH flows can get a fairer performance and better overall QoE. The average stalling time of flows is reduced by 50%, thereby improving the overall QoE significantly. We emphasize that using the “isolation” approach in achieving better performance is not limited to any traffic characteristics and does not assume any buffer size or CC algorithm used. However, it remains challenging to decide on how to isolate the flows and to apportion the bandwidth among them, practically and efficiently. Motivated by this, we further propose a practically scalable solution to isolate each service class’s flows into separate sub-classes by allocating an appropriate amount of bandwidth to them using a so-called (β, γ) -fairness mechanism, as in Figure 1

III. THE DIFFPERF FRAMEWORK

In this section, we present the DiffPerf framework shown in a Figure 1 in a top-down manner. We first describe how DiffPerf allocates bandwidth capacity among the SCs, based on an optimization approach. We will derive closed-form allocation solution and show its feature of guaranteed performance differentiation. We then present a DiffPerf data-driven approach. We show how DiffPerf classifies flows and does *optimization* of bandwidth allocation. We describe a flexible statistical learning method that DiffPerf uses to classify flows, and the bandwidth allocation used by DiffPerf for sub-group isolation.

A. Inter-Class Bandwidth Allocation

We consider an access provider that offers a set $\mathcal{S}$ of service classes over a bottleneck link with capacity C . We denote the set of active flows in any service class $s \in \mathcal{S}$ by $\mathcal{F}_s$ and the cardinality of $\mathcal{F}_s$, i.e., the number of flows in class s , by n_s . To differentiate the performance for flows in different service classes, the access provider needs to allocate appropriate amount of bandwidth to each service class. To accomplish this in a principled manner, we formulate the bandwidth allocation as an optimization of the allocation $\mathbf{X} = (X_s : s \in \mathcal{S})$ that solves a general utility maximization problem as follows.

$$\max_{\mathbf{X}} \sum_{s \in \mathcal{S}} n_s U_s \left(\frac{X_s}{n_s} \right) \quad (1)$$

$$\text{s.t.} \quad \sum_{s \in \mathcal{S}} X_s \leq C \quad \text{and} \quad X_s \geq 0, \quad \forall s \in \mathcal{S}. \quad (2)$$

Under the link capacity constraint (2), the above mathematical program tries to maximize the aggregate utility over all service classes, where for each service class s , it counts the number of flows n_s multiplied by the per-flow utility $U_s(X_s/n_s)$ over the average capacity X_s/n_s allocated to each flow. In particular, we adopt and generalize the well-known weighted α fairness family of utility functions [23] as follows. In this family of utility functions, each service class s will be assigned a weight w_s that indicates the relative importance of the service class, resulting in differentiated per-flow bandwidth allocation across the service classes. By controlling the parameter α , the access provider can express different preferences over various notions of fairness. When α approach 0, the utility tends to be measured purely by the allocated bandwidth; when α approaches $+\infty$, the solution converges to the weighted max-min fair allocation among the flows. In particular, a trade-off of a weighted proportional fair solution can be obtained by solving the optimization problem when α is set to be 1. Thus, besides the differentiation factor w_s among service classes, the service operator can choose the value of α to tradeoff fairness.

Theorem 1. *If an allocation $\mathbf{X}$ maximizes the aggregate utility over all service classes, it must satisfy*

$$X_s = \frac{n_s \sqrt[\alpha]{w_s}}{\sum_{s' \in \mathcal{S}} n_{s'} \sqrt[\alpha]{w_{s'}}} C, \quad \forall s \in \mathcal{S}. \quad (3)$$

Proof. Our bandwidth allocation problem is to maximize a concave objective function subject to linear constraints; thus, this optimization problem amounts to maximizing a concave function over a convex set. From optimization theory, this is a convex optimization problem and maximizing a concave function over a convex set has a unique solution. By Karush-Kuhn-Tucker (KKT) conditions, the unique solution must satisfy

$$\begin{cases} w_s \left(\frac{X_s}{n_s} \right)^{-\alpha} - u + u_G = 0 \quad \text{and} \quad u_s X_s = 0, \quad \forall s \in \mathcal{S}, \\ u \left(\sum_{s \in \mathcal{S}} X_s - C \right) = 0 \end{cases}$$

where u and $(u_s : s \in \mathcal{S})$ are KKT multipliers and satisfy $u, u_s \geq 0$ for any $s \in \mathcal{S}$.

By solving the above equations, we can derive that

$$\begin{cases} X_s = \frac{n_s \sqrt[\alpha]{w_s}}{\sum_{s' \in \mathcal{S}} n_{s'} \sqrt[\alpha]{w_{s'}}} C \quad \text{and} \quad u_s = 0, \quad \forall s \in \mathcal{S}, \\ u = \left(\frac{\sum_{s' \in \mathcal{S}} n_{s'} \sqrt[\alpha]{w_{s'}}}{C} \right)^\alpha. \end{cases}$$

Theorem 1 provides the closed-form solution of the utility maximization problem. Based on the optimal allocation solu-

tion in Equation (3), we derive the ratio of the average per-flow capacities of any two service classes $s, s' \in \mathcal{S}$ as

$$\left(\frac{X_s}{n_s}\right) : \left(\frac{X_{s'}}{n_{s'}}\right) = \sqrt[\alpha]{\frac{w_s}{w_{s'}}}. \quad (4)$$

This result implies that performance differentiation is achieved by enforcing a fixed ratio for the per-flow bandwidth capacity across SCs, which is controlled by the weights $w_s, w_{s'}$ and the fairness parameter α . Equation (4) explicitly shows that the optimal solution effectively allocates a higher average per-flow capacity in the service class that has a larger weight, which is desirable and expected for the better service class. In particular, we also see that when α is set to be 1, the weighted proportional fair allocation leads an average per-flow allocation that are proportional to the weights of the SCs.

B. Intra-Class Bandwidth Allocation

1) *Flow Classification and Isolation*: By relying on QoE as a similarity metric to classify the flows, this choice requires explicit feedback via the receiver to the AP vantage point, which is difficult to afford in practice. We, therefore, want to leverage SDN functionalities to find other metrics to use at the vantage point. The first metric that comes to mind is RTT. However, the use of real-time RTT samples could not be taken solely as indicators of performance issues without other information such as underlying congestion control mechanism, buffer size [20], and packet route security. Even if we make assumptions about the availability of this information, measuring flow RTT at the APs is unreliable. At the SDN control plane, measuring RTT inflates a variable and high RTT based on our measurements in ODL control plane, while measuring it in the SDN data plane [24] may not scale well due to the memory space constraints. Instead, we emphasize that throughput of TCP flows is the appropriate and robust metric that indicates the collective impact of the interaction of network parameters to the user-perceived performance. Next, we show how to utilize the throughput measure as a proxy to determining whether flows are similar to each other and to identify effectively which flows are affected.

Because the number of groups and the number of flows for each group may change and are not known in real scenarios, we adopt general statistical metrics for classification. Given the achieved throughput x_f of the flows $f \in \mathcal{F}_s$ in the SC s , the mean and standard deviation of the flows' throughput are defined as $\bar{x}_s = \frac{1}{n_s} \sum_{f \in \mathcal{F}_s} x_f$ and $\sigma_s = \sqrt{\sum_{f \in \mathcal{F}_s} \frac{(x_f - \bar{x}_s)^2}{n_s}}$.

Because the achieved throughput x_f of each flow depends on the number n_s of competing flows and their characteristics, buffer size, and the allocated capacity X_s that ultimately determines the network congestion imposed on all flows, instead of using absolute throughput thresholds to classify flows, we adopt the following statistical metric that orders and measures the relative throughput values among all flows.

Definition 1. Given the mean $\bar{x}_s$ and standard deviation σ_s , the standard score of a flow f 's throughput is defined by $z_f = (x_f - \bar{x}_s)/\sigma_s$.

When a flow's throughput is above (or below) the mean, its standard score or z -score is positive (or negative, respectively). This z -score captures the signed fractional number of standard deviations by which it is above the mean value, see Figure 3.

Figure 3: Flow classification using statistical (Gaussian) distribution.

Without loss of generality, we divide a set $\mathcal{F}_s$ of flows into two sub-classes: lower sub-class $\mathcal{F}_s^L$ and upper sub-class $\mathcal{F}_s^H$, based on each flow's z -score compared with a pre-defined threshold β , where $\mathcal{F}_s^L(\beta) = \{f \in \mathcal{F}_s : z_f < \beta\}$ and $\mathcal{F}_s^H(\beta) = \mathcal{F}_s \setminus \mathcal{F}_s^L(\beta)$. The set $\mathcal{F}_s^L$ contains the flows that achieved the lowest throughput values (i.e., the negatively affected flows). Thus, our goal is to identify them so that we isolate and allocate the appropriate amount of bandwidth to them accordingly. We use a non-positive value of β to capture flows whose throughput are $|\beta|$ deviations lower than the SC achieved average $\bar{x}_s$. Because the set $\mathcal{F}_s^L$ grows monotonically with the parameter β , i.e., $\mathcal{F}_s^L(\beta_1) \subseteq \mathcal{F}_s^L(\beta_2) \forall \beta_1 < \beta_2$, a smaller value of β makes a more conservative decision on the lowest throughput flows, avoiding mis-classifications. We will further study how the values of β affect the performance of flows in a later section via experimental evaluations.

2) *Bandwidth Allocation Model*: After classifying flows into two sub-groups, we isolate them into two sub-classes and determine how much bandwidth X_s^L and X_s^H to allocate for each sub-class. To fully utilize bandwidth capacity, our solution needs to satisfy $X_s^L + X_s^H = X_s$.

The throughput of some flows might be naturally low and might not be able to achieve the targeted throughput X_s/n_s even allocated that amount of capacity. As a result, enforcing the per-flow allocation of X_s/n_s will result in resource wastage. The key question to answer is how much per-flow capacity we should allocate to the flows $\mathcal{F}_s^L$, whose innate throughputs are less than what is needed to achieve the average throughput $\bar{x}_s$ or to utilize the per-flow allocated capacity X_s/n_s in theory. Since these flows might not be able to achieve the average throughput, the per-flow allocation should be no higher than X_s/n_s . On the other hand, by isolating negatively affected flows from high-throughput flows (i.e., the flows $\mathcal{F}_s^H$ that cause the performance issues of flows $\mathcal{F}_s^L$), we expect them to achieve higher throughput than what are being achieved; and therefore, we should allocate more capacity for the set $\mathcal{F}_s^L$ of flows than their aggregate achieved throughput. To this end, we allocate the average amount of bandwidth capacity for the per flow of set $\mathcal{F}_s^L$ as

$$\frac{X_s^L(\beta, \gamma)}{|\mathcal{F}_s^L(\beta)|} = \gamma \frac{\sum_{f \in \mathcal{F}_s^L} x_f}{|\mathcal{F}_s^L|} + (1 - \gamma) \frac{X_s}{n_s}, \quad (5)$$

where we define the set of flows whose throughput are below the mean $\bar{x}_s$ by $\mathcal{F}_s^- \triangleq \{f \in \mathcal{F}_s : x_f < \bar{x}_s\}$ and introduce a parameter $\gamma \in [0, 1]$ to control the allocated capacity flexibly. In particular, for one extreme of $\gamma = 1$, the solution allocates the average throughput of the set $\mathcal{F}_s^-$ of flows as the per-flow capacity for the lower sub-class $\mathcal{F}_s^L(\beta)$, which must be lower than the average throughput $\bar{x}_s$ and the average capacity X_s/n_s of all flows. In this case, the per-flow capacity allocated for the lower sub-class $\mathcal{F}_s^L$ is lower than that allocated for the upper sub-class $\mathcal{F}_s^H$, under which resource wastage is reduced and resource is utilized more efficiently. For the other extreme of $\gamma = 0$, the solution simply isolates the two sub-classes and equally allocates an average capacity X_s/n_s as the per-flow capacity for both upper and lower sub-classes, under which per-flow fairness is enforced regardless of how efficiently the resource is utilized. Thus, by choosing the value of γ between 0 and 1, we can make a trade-off between resource fairness and utilization. However, this depends on the interaction of TCP algorithm with network buffer size. As opposed to the shallow buffer, the deep buffer allows low-throughput TCP flows (especially those negatively affected due to heterogeneity of RTTs) to stabilize their transfer. Thus, in the vantage point of a deep buffer, if the low-throughput flows were crowded out by others, then they can perform better if $\gamma = 0$. However, this is not the case in a shallow buffer that does not allow negatively affected flows to ramp up quickly and perhaps this even leads to lower utilization.

By Eq. (5), we also have the next theorem showing 1) lower bounds of per-flow capacities re-allocated to the lower and upper sub-classes $\mathcal{F}_s^L$ and $\mathcal{F}_s^H$; and 2) the monotonicity of the average throughput of the flows within $\mathcal{F}_s^L$ and the average per-flow capacity re-allocated to the $\mathcal{F}_s^H$ on the parameter β .

Theorem 2. *Given any fixed parameter γ , for any transport slice or service class $s \in \mathcal{S}$, 1) the average achieved throughput of the flows within the lower sub-class $\mathcal{F}_s^L$ is non-decreasing in β and always no higher than the average per-flow capacity re-allocated to $\mathcal{F}_s^L$; 2) the average per-flow capacity re-allocated to the upper sub-class $\mathcal{F}_s^H$ is non-decreasing in β and always no lower than X_s/n_s .*

Proof. By the definition of the set $\mathcal{F}_s^L(\beta)$, we know that for any two thresholds $\beta_1 < \beta_2$, $\mathcal{F}_s^L(\beta_1) \subseteq \mathcal{F}_s^L(\beta_2)$. For any two flows $\bar{f}$ and f satisfying $\bar{f} \in \mathcal{F}_s^L(\beta_1)$ and $f \in \mathcal{F}_s^L(\beta_2) \setminus \mathcal{F}_s^L(\beta_1)$, we have $x_{\bar{f}} < x_f$ because $z_{\bar{f}} < \beta_1 \leq z_f < \beta_2$. Therefore, it satisfies

$$\begin{aligned} \frac{\sum_{f \in \mathcal{F}_s^L(\beta_1)} x_f}{|\mathcal{F}_s^L(\beta_1)|} &\leq \frac{\sum_{f \in \mathcal{F}_s^L(\beta_1)} x_f + \sum_{f \in \mathcal{F}_s^L(\beta_2) \setminus \mathcal{F}_s^L(\beta_1)} x_f}{|\mathcal{F}_s^L(\beta_1)| + |\mathcal{F}_s^L(\beta_2) \setminus \mathcal{F}_s^L(\beta_1)|} \\ &= \frac{\sum_{f \in \mathcal{F}_s^L(\beta_2)} x_f}{|\mathcal{F}_s^L(\beta_2)|}, \end{aligned}$$

i.e., the average achieved throughput of the flows within the lower sub-class $\mathcal{F}_s^L$ is non-decreasing in β . By Eq. (5), when $\beta = 0$, we have $\frac{\sum_{f \in \mathcal{F}_s^L(\beta)} x_f}{|\mathcal{F}_s^L(\beta)|} = \frac{\sum_{f \in \mathcal{F}_s^-} x_f}{|\mathcal{F}_s^-|} \leq \frac{X_s^L(\beta)}{|\mathcal{F}_s^L(\beta)|}$ because $\mathcal{F}_s^L = \mathcal{F}_s^-$. In other words, the average achieved throughput

of the flows within the lower sub-class $\mathcal{F}_s^L$ equals the average per-flow capacity re-allocated to $\mathcal{F}_s^L$.

Because $|\mathcal{F}_s^L|$ is non-decreasing in β , $\gamma n_s / (n_s - |\mathcal{F}_s^L|)$ is non-decreasing in β . By Eq. (5), the capacity allocated to the per-flow of the upper sub-class satisfies

$$\begin{aligned} \frac{X_s^H}{|\mathcal{F}_s^H|} &= \frac{X_s - X_s^L}{|\mathcal{F}_s^H|} = \frac{\gamma n_s}{n_s - |\mathcal{F}_s^L|} \left(\frac{X_s}{n_s} - \frac{\sum_{f \in \mathcal{F}_s^-} x_f}{|\mathcal{F}_s^-|} \right) \\ &\quad + \gamma \frac{\sum_{f \in \mathcal{F}_s^-} x_f}{|\mathcal{F}_s^-|} + (1 - \gamma) \frac{X_s}{n_s} \geq \frac{X_s}{n_s}. \end{aligned}$$

Thus, it is non-decreasing in β and no lower than X_s/n_s . ■

Theorem 2 states that as the parameter β increases, the average throughput of the flows $\mathcal{F}_s^L$ of the lower sub-class would also increase because more high-throughput flows would be classified into $\mathcal{F}_s^L$. It also tells that this achieved average throughput must be no higher than the per-flow capacity re-allocated to them in $\mathcal{F}_s^L$. This property guarantees our design objective of allocating more capacity for the flows in the lower sub-class than their aggregate achieved throughput. Theorem 2 also states that as β increases, flows within the upper sub-class $\mathcal{F}_s^H$ would be re-allocated more per-flow bandwidth capacity although fewer flows would be classified into the sub-class. Thus, service operators can choose the value of β to control the scales of the sub-classes and both β and γ to control the bandwidth capacity allocated to the flows of the two sub-classes, which we refer to it as (β, γ) -fairness.

Before we close this section, we would like to emphasize that although DiffPerf classifies the flows into two sub-groups for simplicity, its statistical method of classification and the corresponding bandwidth allocation can be applied in a top-down recursive manner to further split any sub-group for a more fine-grained optimization.

IV. IMPLEMENTATION

We implemented two in-network prototypes of DiffPerf; one in OpenDaylight with OpenFlow, and the other on the programmable Tofino switch. Figure 4 depicts the implementation structure of DiffPerf.

A. DiffPerf Prototype on OpenDaylight with OpenFlow

We implement DiffPerf as an application on the popular industry-grade open-source SDN platform—the OpenDaylight (ODL) controller. We particularly develop a native MD-SAL (Model-Driven Service Adaptation Layer) application on ODL which comprises the use of different technologies such as OSGi, Karaf, YANG, blueprint container, and messaging patterns as RPC, publish-subscribe, and data store accesses [25]. We skip implementation details for the sake of brevity. There are five main modules: *Flow Processor*, *TCP session monitoring*, *Statistics Collector*, *Network-Wide Flow Optimization*, and *Bandwidth Enforcer*, that are interconnected as shown in the Figure 4 we briefly describe them below.

Figure 4: DiffPerf implementation.

1) *Flow Processor*: This module determines the pre-defined service class, assigns user-specified service classes to newly joining flows in the network, and assigns the corresponding weight to new flows. We use the YANG modeling language [26] to define the service classes.

2) *TCP session monitoring*: This module carries out regular flow maintenance; i.e., the flow processor inserts new flows into the data store, removes inactive or completed flows, etc.

3) *Statistics Collector*: DiffPerf performs in-network performance optimization. For DiffPerf to work, we need to obtain real-time estimates of the throughput of each active flow as well as link bandwidth. This module pulls out measurement information using event-based handlers from the *operational data store*. The data store, in turn, uses the OpenFlow APIs to request the data plane to report flow measurements; The per-flow measures of interest are packet counts, byte counts, and duration.

4) *Network-Wide Flow Optimization*: This module is the core part of DiffPerf, which is responsible for inter- and intra-class bandwidth optimization described in Section III. The optimizer runs every Δ_t interval, getting input from two modules described above - *Flow Processor* and *Statistics Collector*. While the former provides mapping of flows to user-specified service classes, the latter provides real-time measurements on the active flows in the data plane. Given the input information, the inter- and intra-class optimizers are executed. The output of optimization is given to the *Bandwidth Allocation* module to construct the portion of bandwidth for each SC and for each sub-class with every SC.

5) *Bandwidth Enforcer*: To materialize bandwidth allocation, each sub-class should use its designated bandwidth in an isolated manner. A naive approach to implementing this is to leverage multi-queues at the switch egress port so that each sub-class maps into an isolated queue. However, there are two practical challenges. First, in commodity switches the number of queues at egress port is usually limited to a small number [27], [28], meaning that the number of available queues could be less than the number of flow sub-classes. Second, current OpenFlow switches do not expose APIs to update the weight of the queues dynamically. Without this capability (of dynamically changing queue weights), the bandwidth allocated to queues cannot be changed as and when required. To overcome both limitations, we leverage the *metering* feature available in OpenFlow switches. Instead of defining queues

and updating their bandwidth at egress port, the *Bandwidth Enforcer* module essentially does enforcement at the ingress side of the switch. That is, multiple meters corresponding to the number of sub-classes are defined; and based on the output of the optimizer, the rate of each sub-class is attached to a specific meter dynamically. The OpenFlow APIs encapsulate the allocated bandwidth into OpenFlow messages and install them throughout the *Bandwidth Enforcer*.

B. DiffPerf Prototype with Programmable Data Plane

Further, we implement another prototype of DiffPerf on a custom-made C controller that is connected to Barefoot Tofino switch [29] which allows flexible buffer sizing and fine-grained line-rate telemetry. We particularly implement a statistics building block in the data plane *TCP session monitoring* module to track the number of bytes transmitted by the active flows. Additionally, we re-implement *Bandwidth Enforcer* and *Statistics Collector* modules to fit the run-time APIs exposed by Tofino switch. We also leverage the Tofino switch static APIs to update the weight of the queues dynamically and configure their sizes by the control plane. The remaining modules are kept the same with minor modifications.

V. EXPERIMENTAL EVALUATION

Figure 5: DiffPerf Testbed.

We evaluate DiffPerf by carrying out experiments on a realistic testbed, shown in Figure 5. We describe the details below.

A. Testbed Setup

OpenFlow Brocade switch experiments: We set up a testbed for video streaming between a DASH client (i.e., video player) called `dash.js` and DASH server over an SDN network; Our testbed consists of 12 servers, 10 of which are used to host DASH clients and one each for hosting DASH server and ODL controller. The 10 servers running DASH clients are connected to the DASH server such that they compete (for video segments) at a downstream bottleneck link from a SDN-enabled Brocade ICX-6610 24-port physical switch (i.e., BNG in Figure 5). We evaluated DiffPerf in three different scenarios. In Scenario 1 (Section V-C1) and Scenario 2 (Section V-C2), each physical server hosts up to 4 DASH clients, each client runs in a VM, and all clients are connected to the DASH server over a 50 Mbps downstream bottleneck link. For Scenario 3 (Section V-C3), we scale up number of DASH clients—each physical server hosts 15

DASH clients, all run as docker containers and connected to DASH server over a 200 Mbps downstream bottleneck link.

Intel Tofino switch experiments: The experiments with Tofino programmable switch (i.e., BNG in Figure 5) concentrate on evaluating the impact of buffer size on the performance of bottlenecked flows and how DiffPerf enables switch buffer to perform better (i.e., improve the overall flow performance). The evaluations are carried out with multiple switch buffer sizes: 100KB, 1MB, and 10MB. Tofino exposes a set of APIs for traffic manager applications to manage buffer allocation from both ingress and egress ends. We use `bf_tm_q_app_pool_usage_set` API to aid in setting buffer size for the queues of the egress port attached to the bottlenecked link. Buffer size is specified in terms of cells, where each cell size is 80 bytes. The buffer precedes a 120 Mbps bottleneck link that transfers video segments from DASH server to 100 DASH clients. The results are presented under the last part of Scenario 2 (Section V-C2).

Except for Scenario 1 (Section V-C1) and the first part of Scenario 3 (Section V-C3), assuming majority of flows in the Internet have short RTTs [30], we partition the clients into two sets in 70:30 ratio based on the RTT_{min} values configured: the mean and standard deviation of the bigger set are 64ms and 16ms, respectively, and that of the other are 224ms and 32ms, respectively. We use the network emulator *netem* [31] at the server machines running the DASH clients to set the latency. For streaming, we use the Big Buck Bunny video sample that lasts for 600 seconds and has been encoded into 3 bitrate levels—1.2 Mbps, 2.2 Mbps and 4.1 Mbps—of equal segments (i.e., each segment is 2 seconds long). Thus, a DASH client can choose the bitrate levels and segments for streaming video, based on the measured congestion level of the network. We compare DiffPerf against two most popular TCP congestion control algorithms on the Internet [32]; TCP CUBIC [33] and TCP BBR [34].

B. Metrics for evaluation

To evaluate the performance of DiffPerf and TCP variants, we use two metrics. One is per-flow average throughput, which in our case corresponds to the average throughputs of all DASH client. The other metric of importance is the user-perceived quality-of-experience (QoE). The QoE metric is adopted based on the widely used model proposed by [35], and is expressed as: $QoE = \sum_{n=1}^N q(R_n) - \lambda \sum_{n=1}^{N-1} |q(R_{n+1}) - q(R_n)| - \mu T_{stall} - \mu_s T_s$. This QoE definition uses various performance factors such as the average playback bitrate R_n over the total N segments of the video, the average variability of the consecutive segments bitrate represented by the second summation, the duration of rebuffering T_{stall} (i.e., the duration of time the player's playout buffer has no content to render), and startup delay (T_s) (i.e., the lag between the user clicking and the time to begin rendering). As in [36][35], q maps a bitrate to a quality value; λ is usually set to one, μ and μ_s are set to the maximum bitrate of the video sample. We measure QoE for the entire duration of the video. We use the terms flow, user, and client interchangeably.

Figure 6: The Achieved throughput and corresponding QoE of multiple service classes for different values of α .

C. Experiments Results

1) *Scenario 1: Evaluation of inter-class performance:* In this scenario we evaluate DiffPerf's inter-class bandwidth allocation model. We assume the access provider offers three classes of services: Golden (G), Silver (S) and Bronze (B), with weights of 3, 2 and 1, respectively. We run a set of experiments to evaluate the bandwidth allocated to users of different classes under different values of α . We assign 13 DASH clients to each service class, thereby having a total of 39 DASH clients in this scenario. All flows experience homogeneous RTT in this scenario.

Figures 6(a) plots, for each SC, the average throughput achieved by all flows in that class, for different values of α . Evidently, the ratios of the estimated average throughput of flows across the service classes closely follow the ratios obtained from our model (refer Eq. 4). In addition, the average throughput is converging with increasing α .

Figure 6(b) plots the average QoE of all flows in each SC. Observe that the QoE of service class B is low, when the average throughput achieved (given in Figure 6(a)) is low. The QoE of the three service classes converge with increasing value of α . With increasing α , as resources would be fairly shared among the competing flows, it is expected that higher level QoE also reflects this fair sharing given that the flows have homogeneous RTTs.

2) *Scenario 2: Evaluation of intra-class performance:* In this part, we evaluate our proposed performance-aware fairness, (β, γ) -fairness. That is, DiffPerf's β capability to mitigate the bias brought against the affected flows by the interaction between TCP CUBIC, TCP BBR, flows with heterogeneous RTTs, and switch buffer size. Also, we present the flexibility of γ in enabling a feature of practical interest, the trade-off between network efficiency and flow fairness. Recall, DiffPerf uses statistical flow classification and bandwidth allocation to the classified sub-classes to appropriately allocate a higher capacity to the negatively affected flows.

DiffPerf based on CUBIC. In Figure 7 we present the aggregate throughput achieved by DASH clients belonging to the two different sub-classes, under TCP CUBIC, TCP BBR, and DiffPerf over CUBIC. We run experiments for different values of β . For better clarity, we show the results only for $\gamma = 0.5$. The number of flows classified into the lower sub-class based on our statistical classification, for $\beta = 0, -0.25, -0.5, -0.75$, and -1.0 , are 22, 17, 12, 9, and

Figure 7: Aggregate throughput of the sub-classes flows.

Figure 8: QoE of the sub-classes flows.

5, respectively. The figure shows that DiffPerf's *isolation* enables the DASH clients in the lower sub-class to achieve higher throughput than both TCP variants, while also being able to achieve comparable aggregate throughput as TCP². And β gives an access provider flexibility to decide on the flow classification based on a simple and intuitive metric (z -score). While a small number of flows get classified in the lower sub-class, we note that these were also the worst affected ones. Figures 8(a) and (b) plot the QoE achieved by each DASH client, for $\beta = -0.25$; here we plot for different values of γ as well. The former figure plots QoE of the upper sub-class of flows, the latter plots the same for the lower sub-class of flows. The key observation is that, the flows in the lower sub-class perceive higher QoE under DiffPerf than under both TCP variants, and only at the cost of a small number of flows in the upper sub-class. Another observation is that, with lower γ , DiffPerf gives fairer QoE to the clients; DiffPerf with $\gamma = 0$ is the most fair. DiffPerf not only improves the fairness, but we also calculated the overall QoE values for all flows; and it shows that DiffPerf, via flow isolation as well as performance-aware bandwidth allocation, improves significantly the overall QoE compared to TCP solely. It performs 1.86, 1.62, and 1.58 times higher QoE than CUBIC, at $\gamma = 0, 0.5$, and 1, respectively; Similarly, it performs 1.63, 1.42, 1.38 times higher QoE than BBR.

Fairness-efficiency tradeoff: We run experiments for different values of γ , to analyze the trade-off between efficiency (i.e., bandwidth utilization) and fairness (achieved throughput among the flows or clients), β is set to -0.25 . Figure 9 depicts that the aggregate throughput increases as γ increases. Meanwhile, the average throughput of the lower sub-class decreases, and that of the upper-class increases, as γ is increased from 0

Figure 9: DiffPerf fairness-efficiency tradeoff.

to 1 (refer to the second Y-axis). Evidently, the parameter γ affects the average flow throughput of both sub-classes. When γ approaches 0, the flows are allocated equal bandwidth at both sub-classes; hence the fairness is increased, but because some flows in the lower sub-class are naturally limited to low throughput, thus the utilization is decreased. Conversely, when γ approaches 1, DiffPerf allocates higher bandwidth to the upper sub-class that likely contains flows with a greater tendency to exploit provisioned bandwidth. Hence, this results in better bandwidth utilization but the fairness between the flows in the sub-classes is decreased.

DiffPerf based on BBR. As TCP BBR has recently gained wide-spread attention, DiffPerf is evaluated over TCP BBR. Figure 10 shows that DiffPerf's *isolation* enables DASH clients in the lower sub-class (i.e., the affected flows) to achieve higher throughput than BBR, while also being able to achieve comparable aggregate throughput as BBR. The number of flows classified into the sub-classes is illustrated next to DiffPerf bar. For example, at $\beta = 0$, 26 flows are classified into lower sub-class, and 14 flows to upper sub-class. Figure 11(a) and (b) plot the perceived QoE of DASH clients in the aforementioned sub-classes. The lower-subclass flows with DiffPerf perceive better QoE than with BBR.

Figure 10: Aggregate throughput of the sub-classes flows.

Figure 11: QoE of the sub-classes flows.

²Notice that all CUBIC, BBR, and DiffPerf achieve aggregate throughput slightly higher than 50Mbps; this is due to the burstiness tolerated by OpenFlow meters. Also, BBR achieves the highest aggregate as it fundamentally does not passively react to packet loss or delay as signals of congestion.

Impact of buffer size: Based on the experiments carried

Figure 12: The clients' QoE under various network conditions (i.e., multiple buffer sizes, variable RTTs, and different TCPs).

out on the Tofino switch, this part presents the impact of buffer size on the performance of bottlenecked flows under CUBIC and BBR as shown in Figure 12 and the flow optimization achieved using DiffPerf shown in Figures 15 and 16. Note, Tofino switch updates DiffPerf with flow statistics every 1s (i.e., the sampling rate). At every interval of $\Delta_t = 5s$, DiffPerf uses the last measured statistics for regulating network flows in the next immediate time interval. Figures 14 and Figure 16 compare the stalling time and QoE between BBR and DiffPerf over BBR (with $\beta = -0.25$ and $\gamma = 0$), respectively.

With a shallow buffer (i.e., 1MB), BBR achieves three times higher QoE than with a deep buffer (i.e., 10MB). However, DiffPerf achieves 1.58 times higher QoE than BBR with shallow buffer and 2.6 times higher QoE than BBR with deep buffer. We note that the shallow buffer leads to overall better user-perceived quality. However, quality worsens with a much smaller buffer size (i.e., 100KB). The deep buffer might help low-throughput flows, especially those affected by the interaction of TCP with flow RTT, to achieve better QoE but it increases packet queuing delay. The very shallow buffer (i.e., 100KB) reduces packet queuing delay but increases packet losses. Hence, both these extreme buffer sizes increase the DASH client's average stalling time (i.e., the duration of time the player's playout buffer has no content to render). As in Figure 14 the client average stalling time under BBR is 92.2, 56.4, and 76.5 seconds, while under DiffPerf, the client average stalling time is reduced to 38.8, 28.8, and 58.4 seconds over buffer sizes 100KB, 1MB, and 10MB, respectively. DiffPerf thus demonstrates to be effective in improving the user-perceived quality for multiple buffer sizes, as shown in Figure 16. Additionally, we run same experiments for CUBIC and DiffPerf over CUBIC (with $\beta = -0.25$ and $\gamma = 0$). Although TCP CUBIC outperforms TCP BBR in the default settings, DiffPerf still performs better than CUBIC. Figure 13 shows also that DiffPerf reduces total stalling time in all cases of experimented switch buffer sizes, compared to CUBIC. Thus, DiffPerf achieves fairer QoE (and overall higher QoE) than CUBIC as illustrated in Figure 15. Lastly, it is worth noting that from this set of experiments, the buffer size of 1MB makes a better trade-off between queuing delays and packet losses. Overall, DiffPerf is fairer than both CUBIC and BBR in terms of client's throughput, client's QoE, and provides the highest overall QoE.

3) *Scenario 3: The Dynamics of DiffPerf*: Finally, to understand how DiffPerf performs in real-world cases, we evaluate it in two dynamic environments where users from different service classes join and leave the network at different times. In this set of experiments, we conduct the evaluation based on SDN OpenFlow network centralized by ODL controller, where 150 DASH clients share a 200 Mbps bottleneck link. At every interval of $\Delta_t = 15s$, DiffPerf uses the last measured statistics, such as number of active flows and each flow's instantaneous throughput ($\delta = 0$), to subsequently send command to the switch for regulating network flows in the next immediate time interval. OpenFlow switch (i.e., BNG in Figure 5) updates DiffPerf with flow statistics every 3s (i.e., the default sampling rate in Brocade ICX-6610 switch).

Homogeneous environment: In future enterprise and operator networks, the homogeneity of flows are possible with proliferation of CDNs and the shift towards edge computing. Hence, we evaluate the adaptive capabilities of DiffPerf to differentiate the performance between service classes based on active flows of homogeneous characteristics. Therefore, we set similar RTTs for all flows (i.e., DASH clients). The arrivals of the DASH client requests follow the Poisson process with rate $\lambda = 0.5$ client/s. A client exits after the entire video (that lasts for 600 seconds) is streamed. The DASH clients subscribe to G, S, and B, service classes in the ratio 1:1:1. The weights of service classes are kept the same as before; i.e., G:S:B = 3:2:1. We set the value of α to 1 to achieve weighted proportional share of bandwidth among the service classes.

Figure 17 shows the number of active DASH flows in the three service classes. Although the video being streamed is of 600 seconds, observe that the G-class and S-class client complete earlier than B-class. This is true because flows (i.e., dash.js players) that are provisioned with higher bandwidth have a higher chance to play out video streams much faster. Figure 18 plots the dynamic bandwidth allocation recommended by DiffPerf, for each service class, which accounts for number of active flows in each service class. It shows that DiffPerf adapts quickly to the departure of flows, allocating the spare capacity to the remaining active flows. The corresponding achieved throughput of the allocated bandwidth and the average throughput achieved by flows are illustrated in Figure 19 and Figure 20, respectively. In steady state when number when all flows are active in all service classes, the total bandwidth and throughput appear proportional to the weight of service classes; G:S:B = 3:2:1. This demonstrates that utility function of DiffPerf in the macroscopic level is able to realize the desired performance of the service operator who wants to apportion the bandwidth proportionally. Note that throughput oscillations is mainly caused by DASH clients at the application layer. Lastly, we plot the CDF of the user-perceived quality for all service classes, in Figure 21. Clearly, B-class clients perceive the worst quality as they are not provisioned with sufficient bandwidth to constantly play out even the lowest bitrate quality.

Heterogeneous environment: In this set of experiments, DiffPerf performs in two levels. At the upper level, it first

Figure 13: The total stall time for multiple buffer sizes of CUBIC and DiffPerf.

Figure 14: The total stall time for multiple buffer sizes of BBR and DiffPerf.

Figure 15: The QoE for multiple buffer sizes of CUBIC and DiffPerf.

Figure 16: The QoE for multiple buffer sizes of BBR and DiffPerf.

Figure 17: Service classes cumulative sum of active flows.

Figure 18: Dynamic service classes bandwidth allocation.

performs inter-class performance differentiation. At the lower level, it then performs intra-class optimization based on (β, γ) performance-aware mechanism. In this environment, the

flows have variable RTTs, with the ratio and distribution same as in the scenario 2 in [V-C2](#). The arrivals of the DASH client requests follow the Poisson process with rate $\lambda = 1$ client/s. A

Figure 19: Dynamic service class throughput.

Figure 20: Average DASH client flow throughput.

Figure 21: CDF of QoE for service class flows.

client exits after the entire video (that lasts for 600 seconds) is streamed. The DASH clients subscribe to G, S, and B, service classes in the ratio 1:2:3. The weights of service classes are kept the same as before; i.e., $G:S:B = 3:2:1$. We set the values of α , β and γ to 1, -0.25 , and 0.5 , respectively.

Figure 22(a) shows the arrival and departure of DASH flows. The number of active flows in the two sub-classes, for each of the services classes, are depicted in Figures 22 (a), (b), and (c). Although the video being streamed is of 600 seconds, observe that the G-class clients complete earlier than S-class and B-class; and this is true for both lower and upper sub-classes. Similarly, S-class flows finish earlier than B-class flows. We observe a sudden decrease in the active flows a few times (the dips on the curves); this is not because the flow(s) actually leave the system, rather, due to the expiry of *idle timeout* of flows. When a DASH client stalls or does not send packets (noticed mostly in long RTT clients), the timeout causes the flow to be deemed as an inactive flow. However, once the flow resumes, *DiffPerf* promptly counts it as an active flow. Figure 23 plots the dynamic bandwidth allocation recommended by *DiffPerf*, for each service class and the sub-classes within. The bandwidth allocated accounts for number of active

flows in each service class and their achieved throughput (which is dependent on the RTT_{min}). The bandwidth allocated accounts for number of active flows in each service class and their achieved throughput, optimized via (β, γ) performance-aware mechanism. It shows that *DiffPerf* adapts quickly to the departure of flows (observe time period after 600 seconds), allocating the spare capacity to the remaining active flows. The corresponding throughput realized for this allocation is illustrated in Figure 24. The total throughput for G-class appears lower than other classes as there are lesser clients in G-class than in S-class and B-class. Therefore, the average throughput achieved by flows in G-class is higher than that of S-class and B-class flows, as shown in Figure 25. It can also be observed that, generally, the average throughput achieved by lower sub-class decreases from G-class to S-class to B-class.

The root cause of the throughput oscillations is not only due to TCP dynamics, but also due to the inherent behavior of DASH clients at the application layer, when they compete for the bandwidth. DASH encodes a video content into different levels of bitrate and delegates the selection logic of the bitrate to the client. Depending on the estimation of the bandwidth, a DASH client reacts by switching between different bitrate levels, thus also contributing to oscillations in achieved throughput. However, the recent proposals in the DASH bitrate adaptation logic attempt to dampen these oscillations and stabilize the client's selections of bitrates [35], [36]. Lastly, we measure average QoE for all service classes. We observed that G-class clients perceive better quality, face less stall time. G-class achieves an average QoE of 2.85 and 5 times higher than S-class and B-class, respectively, while S-class is 1.78 times higher than B-class. Meanwhile, as B-class clients are not provisioned with enough bandwidth (considering the number of clients subscribed to B-class, which is three times those subscribed to G-class), B-class experiences 45% more stalling time than G-class during streaming, resulting in low QoE. Thus, these results confirm *DiffPerf*'s capability of performance differentiation in real-world dynamic scenario.

D. Deployability

DiffPerf operations on both OpenFlow and Tofino switches are split across controller and data plane. The controller collects aggregate real-time statistics of the active flows, performs the optimization, and reacts to data plane regularly. The data plane tracks the number of bytes transferred by the active flows. *DiffPerf* does not impose high rate of sampling, which may lead to inaccurate statistics, especially when the DASH clients enter OFF period. Hence, the communication (between controller and data plane) is only at the scale of seconds, and this works well for long running video flows over the Internet. This is also demonstrated by our experimental results. Furthermore, *DiffPerf* can leverage recent techniques such as NetSeer [37] to optimize the overhead of reported statistics to the controller by compression and an appropriate encapsulation. Also, *DiffPerf* is portable, it can be packaged as virtual network function (VNF), associated

Figure 22: DASH clients arrival-departure and service classes cumulative sum of active flows

Figure 23: Dynamic service classes bandwidth allocation

Figure 24: Dynamic service class throughput

Figure 25: Average DASH client flow throughput

with virtual switch datapath, and accelerated using SmartNIC, towards high-performance server-based networking.

VI. RELATED WORK

A. Service Differentiation

Service differentiation is at the core of network quality of service (QoS) provisioning to serve traffic from multiple classes over the network [38], [39], [40], [41]. Two popular approaches have been recognized as a dedicated QoS architectures: Integrated Services (IntServ) and Differentiated

Services (DiffServ). While *IntServ* [40] did not find adoption in the Internet, *DiffServ* [38] inspired a body of work on providing differentiated services. Despite that DiffServ establishes step forward towards service differentiation, it lacks the implementation details. DiffServ does not demonstrate fine-grained differentiation mechanism or adaption based on network state. Many of solutions based on DiffServ mandate a manual configuration of QoS knobs on a per service class basis. Alternatively, *DiffPerf* incorporates varying levels of sophistication in performance differentiation. *DiffPerf*

implements a utility function based framework to enable service operator to practically specify number of service classes and make good balance between the bandwidth share and performance. It also executes a lightweight learning algorithm to further improve the performance on a service class level.

In [42], authors proposed an approach for rate-delay (RD) differentiation by maintaining two queues at the router's output link. While the aspirations resemble DiffPerf, it is still best-effort and does not promise any rate or loss guarantees. [12] discussed a static service differentiation framework for ISP. In short, class-based traffic control and service differentiation have been largely limited to theoretical analyses [42], [43], [44], [45], and have not been experimented on hardware switches with real application traffic. DiffPerf's inter-class utility is general enough to make trade-offs among desirable performance metrics and operates dynamically based on active users and available bandwidth.

B. Bandwidth Allocation

The amount of bandwidth allocated for each flow or the flow's buffer occupancy at the last-mile is affected by algorithms at the host-level (e.g., TCP), port-level buffer sizing and management of network device (e.g., AQM), or both.

1) *Congestion Control (CC) Algorithms*: An increase of network bandwidth also saw the emergence of 'high-speed' CC variants such as FAST [46], BIC [47], CUBIC [33] and BBR [34] for exploiting the benefits of high speed-links with acceptable end-to-end delay. These algorithms are delegated at the end-host and often assume simple tail dropping buffers. This leads to TCP's inability to fairly share the bandwidth of flows with heterogeneous RTTs—a problem known to the community for around two decades [48], [49]—still persists. As demonstrated by our experiments (and also other works, e.g., [50]) CUBIC exhibits such behavior, and so does BBR [20], [51]. This unfairness in achieved throughput worsens when flows with different CC algorithms compete [22]. Another interesting observation from literature is that the relative performance degradation in throughput can be due to more than the single factor of RTT, particularly with the introduction of 5G and FTTH technologies that result in a much higher load of heterogeneous traffic. In this context, we emphasize that DiffPerf is agnostic to the specific RTT of flows and other router specifications (e.g., buffer size) in performing optimization and enforcement of the computed optimized bandwidth; indeed DiffPerf classifies and isolates flows of dissimilar characteristics solely based on tracking their achieved throughput.

2) *Buffer Sizing and Management*: What optimal network buffer size can be acceptable and the possibility of tailoring the use of buffer using a management algorithm (i.e., AQM), to optimize the performance perceived by the user are problems that received significant attention and a lot of debate, though remain unsolved completely to date [20], [52], [21]. Many studies about buffer sizing aim at achieving good utilization with reasonable queuing delay. However, most of them assume a tail drop buffer in the bottleneck and homogeneous traffic

such as flows having the same RTTs. On the other hand, many AQM algorithms assume a specific CC algorithm and only consider the queue length or the queue delay and is thus not aware of the buffer contents. More recent AQM applying per-flow buffer limits to protect the buffer from unresponsive flows to achieve fairness among the flows [53], [54], [55], [56], [57]. For instance, FQcode [54], a recent AQM discipline, allocates a bucket per flow and serves them in a round-robin. However, large memory still has a high impact on the cost of the data plane device. Hence it is practically infeasible to accommodate a very large number of buckets for hashing a large number of flows. Furthermore, there has been a plethora of recent solutions on flow scheduling of data center applications, that is driven based on the intuition of segregation of flows into short and long. However, these solutions cannot be applied for the Internet traffic that is more heterogeneous than the data center in terms of CC, flow RTTs, buffer sizes, ..etc. On contrary, DiffPerf is a data-driven solution that observes and optimizes the performance on top of several interactive system parameters such as buffer sizes, heterogeneous flow patterns, and CC algorithms and is not limited to any of them. Because it classifies flows into sub-groups and requires no training, it is thus scalable and lightweight.

C. User Quality of Experience (QoE)

In the context of video streaming, several approaches are proposed to improve user QoE [36], [14] or to achieve QoE fairness [58]. These approaches attempt to improve the ABR algorithms in the DASH reference player at application layer, based on several performance metrics seen in the application. Our work differs from them in that we propose bottom-up optimization. DiffPerf reacts to the interplay between several network's inherently coupled parameters by continuously improving affected traffic flows. This in large part improves the performance metrics (e.g., QoE fairness) at the application.

VII. CONCLUSION

We propose DiffPerf system that represents a significant step forward in network softwarization and implement an agile and dynamic in-network bandwidth allocation in an ISP-centric settings. DiffPerf offers Internet access providers new capabilities for performance guarantees by dynamically allocating bandwidth to service classes through which the trade-off between fairness and differentiation can be made. In each service class, DiffPerf smartly optimizes flows, particularly the negatively affected as a result of the interplay between several network's inherently coupled parameters such as flow characteristics, network buffer size, and congestion control algorithm. Our experiment results on hardware platforms using real-world video transmission confirm DiffPerf's capabilities of QoE provisioning among different service classes, as well as achieving better QoE fairness and improving overall user-perceived QoE within the same service class, compared to CUBIC and BBR, the two most popular TCP protocols on the Internet.

REFERENCES

- [1] "GSMA Mobile Economy, GSMA Alliance, London., U.K." https://www.gsma.com/mobileeconomy/wp-content/uploads/2021/07/GSMA_MobileEconomy2021_3.pdf, acc. June. 20201.
- [2] "Open Connect," <https://openconnect.netflix.com/en/>, acc. Aug. 2020.
- [3] R. Kundel, J. Wallerich, W. Maas, L. Nobach, B. Koldehofe, and R. Steinmetz, "Queueing at the telco service edge: Requirements, challenges and opportunities," in *Wksp on Buffer Sizing*, 2019.
- [4] P. Faratin, D. D. Clark, S. Bauer, and W. Lehr, "Complexity of Internet interconnections: Tech., incentives and implications for policy," 2007.
- [5] "MEC in 5G networks," https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf, acc. July. 2021.
- [6] K. Bollen and R. Lennox, "Conventional wisdom on measurement: A structural equation perspective." *Psychological bulletin*, 2018.
- [7] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," in *Proc. ACM SIGCOMM*.
- [8] W. Bai, K. Chen, H. Wang, L. Chen, D. Han, and C. Tian, "Information-agnostic flow scheduling for commodity data centers," in *Proc. NSDI*.
- [9] M. Apostolaki, L. Vanbever, and M. G., "Fab: Toward flow-aware buffer sharing on programmable switches," in *Wksp on Buffer Sizing*, 2019.
- [10] "Cisco Annual Internet Report (2018–2023)," <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf>, acc. Aug. 2020.
- [11] "Canopus Networks," <https://canopusnet.com/insights/>, acc. May. 2020.
- [12] V. Sivaraman, S. C. Madanapalli, H. Kumar, and H. H. Gharakheili, "Opentd: Open traffic differentiation in a post-neutral world," in *Proc. ACM SOSR*, 2019.
- [13] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Participatory networking: An api for application control of sdns," 2013.
- [14] A. Bentalab, A. C. Begen, and R. Zimmermann, "Snddash: Improving qoe of http adaptive streaming using software defined networking," in *Proc. ACM MM*, 2016.
- [15] W. Aljoby, X. Wang, D. M. Divakaran, T. Z. Fu, and R. T. Ma, "DiffPerf: An In-Network Performance Optimization for Improving User-Perceived QoE," *Proc. IEEE NetSoft*, 2021. [Online]. Available: <https://www.dropbox.com/s/37dtxpigtxx1kqn/NetSoft2021.pdf?dl=0>
- [16] J. Jaffe, "Bottleneck flow control," *IEEE TOC*, vol. 29, no. 7, pp. 954–962, 1981.
- [17] S. Shenker, "Fundamental design issues for the future internet," *IEEE JSAC*, 1995.
- [18] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A resource allocation model for qos management," in *Proc. IEEE RTSS*, 1997, pp. 298–307.
- [19] Z. Cao and E. W. Zegura, "Utility max-min: An application-oriented bandwidth allocation scheme," in *Proc. IEEE INFOCOM*, 1999, pp. 793–801.
- [20] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of BBR congestion control," in *Proc. IEEE ICNP*, 2017.
- [21] M. Mathis and A. McGregor, "Buffer sizing: a position paper," in *Wksp on Buffer Sizing*, 2019.
- [22] J. Rüth, I. Kunze, and O. Hohlfeld, "An Empirical View on Content Provider Fairness," in "Proc. TMA, 2019.
- [23] R. Srikan, *The mathematics of Internet congestion control*. Springer Science & Business Media, 2012.
- [24] X. Chen, H. Kim, J. M. Aman, W. Chang, M. Lee, and J. Rexford, "Measuring tcp rtt in the data plane," in *Wksp on SPIN*, 2020.
- [25] "OpenDaylight Project," <http://wiki.opendaylight.org>, acc. Jul. 2019.
- [26] M. Björklund *et al.*, "Yang - a data modeling language for the netconf protocol," <https://tools.ietf.org/html/rfc6020>, acc. Aug 2020.
- [27] "Cisco nexus 5500 series nx-os quality of service configuration guide," https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5500/sw/qos/7x/b_5500_QoS_Config_7x.html, acc. Aug. 2020.
- [28] "Brocade ICX 6610 Switches," <https://www.dataswitchworks.com/datasheets/switches/brocade-icx-6610-ds.pdf>, acc. Aug. 2020.
- [29] "Barefoot Networks, Tofino," <https://goo.gl/cdEK1E>, acc. Aug. 2020.
- [30] R. Farahbaksh, A. Cuevas, A. M. Ortiz, X. Han, and N. Crespi, "How far is Facebook from me? Facebook network infrastructure analysis," *IEEE Communications Magazine*, 2015.
- [31] "Network emulation with netem," <https://man7.org/linux/man-pages/man8/tc-netem.8.html>, acc. Aug 2020.
- [32] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The great internet tcp congestion control census," in *Proc. ACM on MACS*, 2019.
- [33] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS operating systems review*, 2008.
- [34] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *Queue*, 2016.
- [35] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM SIGCOMM*, 2015.
- [36] P. K. Yadav, A. Shafiei, and W. T. Ooi, "Quetra: A queuing theory approach to dash rate adaptation," in *Proc. ACM MM*, 2017.
- [37] Y. Zhou, C. Sun, H. H. Liu, R. Miao, S. Bai, B. Li, Z. Zheng, L. Zhu, Z. Shen, Y. Xi *et al.*, "Flow event telemetry on programmable data plane," in *Proc. ACM SIGCOMM*.
- [38] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated service," <http://www.ietf.org/rfc/rfc2475.txt>, acc. Aug 2020.
- [39] X. Wang and H. Schulzrinne, "Pricing network resources for adaptive apps in a diff. services network," in *Proc. IEEE INFOCOM*, 2001.
- [40] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture," <http://www.ietf.org/rfc/rfc1633.txt>, acc. Aug 2020.
- [41] J. Rao, Y. Wei, J. Gong, and C.-Z. Xu, "Qos guarantees and service differentiation for dynamic cloud apps," *IEEE Tran. on NSM*, 2012.
- [42] M. Podlesny and S. Gorinsky, "Rd network services: differentiation through performance incentives," *Proc. ACM SIGCOMM*, 2008.
- [43] B. Han, V. Sciancalepore, D. Feng, X. Costa-Perez, and H. D. Schotten, "A utility-driven multi-queue admission control solution for network slicing," in *Proc. IEEE INFOCOM*, 2019.
- [44] Y.-W. E. Sung, C. Lund, M. Lyn, S. G. Rao, and S. Sen, "Modeling and understanding end-to-end class of service policies in operational networks," in *Proc. ACM SIGCOMM*, 2009.
- [45] M. Zou, R. T. Ma, X. Wang, and Y. Xu, "On optimal service differentiation in congested network markets," *IEEE/ACM Tran. on Net.*, 2018.
- [46] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *IEEE/ACM Tran. on Net.*, 2006.
- [47] Lisong Xu, K. Harfoush, and Injong Rhee, "BIC for fast long-distance networks," in *Proc. IEEE INFOCOM*, 2004.
- [48] C. Barakat, E. Altman, and W. Dabbous, "On TCP performance in a heterogeneous network: a survey," *IEEE Comm. Magazine*, 2000.
- [49] Y. Li, D. Leith, and R. N. Shorten, "Experimental Evaluation of TCP Protocols for High-Speed Networks," *IEEE/ACM Tran. on Net.*, 2007.
- [50] D. Miras, M. Bateman, and S. Bhatti, "Fairness of High-Speed TCP Stacks," in *Proc. AINA*, 2008.
- [51] S. Ma, J. Jiang, W. Wang, and B. Li, "Towards rtt fairness of congestion-based congestion control," *CoRR*, 2017.
- [52] B. Spang, B. Walsh, T.-Y. Huang, T. Rusnock, J. Lawrence, and N. McKeown, "Buffer sizing and video qoe measurements at netflix," in *Wksp on Buffer Sizing*, 2019.
- [53] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. ACM SIGCOMM*, 2002.
- [54] T. Hoeiland-Joergensen, P. McKeeney, D. Taht, J. Gettys, and E. Dumazet, "The flow queue-codel packet scheduler and active queue management algorithm," *IETF Draft, March*, 2016.
- [55] J. C. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Tran. on Net.*, 1997.
- [56] M. Hedayati, K. Shen, M. L. Scott, and M. Marty, "Multi-queue fair queueing," in *Proc. NSDI*, 2019.
- [57] A. Sivaraman, K. Winstein, S. Subramanian, and H. Balakrishnan, "No silver bullet: extending sdn to the data plane," in *Proc. ACM Wksp on Hotnets*, 2013.
- [58] P. Georgopoulos, Y. Elkhathib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide qoe fairness using openflow-assisted adaptive video streaming," in *Proc. ACM SIGCOMM Wksp on FhMN*, 2013.

Walid Aljoby received the B.Sc. degree (Hons.) in computer information systems from the Hashemite University in 2008 and the M.Sc. degree in computer engineering from the Jordan University of Science and Technology in 2013, and the Ph.D degree in computer science from the National University of Singapore (NUS) in 2020. He is currently a Post-doctoral Research Fellow with the department of computer science, NUS and adjunct research fellow with Advanced Digital Sciences Center, the research center of the University of Illinois at Urbana-Champaign in Singapore.

His research interests are broadly in networking and systems, and more specifically on software-defined networking, programmable dataplanes, cloud networking, and network security.

Richard T. B. Ma (SM'16) received the B.Sc. degree (Hons.) in computer science and the M.Phil. degree in computer science and engineering from the Chinese University of Hong Kong, in 2002 and 2004, respectively, and the Ph.D. degree in electrical engineering from Columbia University, in 2010. During the Ph.D. degree, he was a Research Intern with the IBM T. J. Watson Research Center, NY, USA, and the Telefonica Research, Barcelona. He is currently an Assistant Professor with the Department of Computer Science, National University

of Singapore. His current research interests include distributed systems and network economics. He is a co-recipient of the Best Paper Award in the IEEE Workshop on Smart Data Pricing 2015, the IEEE ICNP 2014, and the IEEE IC2E 2013.

Xin Wang received his B.Eng degree in computer science and technology and the Ph.D. degree in computer software and theory from University of Science and Technology of China (USTC), in 2012 and 2017, respectively. He is currently a Research Fellow with the Department of Computer Science, National University of Singapore. His current research interests include distributed systems and Internet economics.

Dinil Mon Divakaran (Senior Member, IEEE) is a Security Research Lead within the CTO Office of Trustwave (a Singtel Company). He is also an Adjunct Assistant Professor of the School of Computing at the National University of Singapore (NUS). Prior to this, he was the Deputy Head of the Network Security Department, at the A*STAR Institute for Infocomm Research. He leads and manages multiple research and development projects within the NUS-Singtel Cyber Security Research and Development Corporate Laboratory based in

NUS. His research experience cuts across both industry and academia. He previously held faculty position with the IIT Mandi, India, where he joined as an Assistant Professor after his Ph.D. He carried out his doctoral studies with ENS Lyon, Lyon, France, within the joint lab of INRIA and Bell Labs. His research interests revolve around the broad areas of network and system security, security analytics (application of stochastic models, machine learning, and deep learning in security domain), protocol analysis, and QoS provisioning (using programmable data planes).

Tom Z. J. Fu obtained the B.Eng degree in Information Engineering from Shanghai Jiao Tong University in 2006. He received the M.Phil and the Ph.D degrees from the Department of Information Engineering at the Chinese University of Hong Kong in 2008 and 2013, respectively. He is currently leading the network transmission algorithm team of Bigo Technology. Before joining Bigo, he was a senior research scientist and analytics area programme manager of the Advanced Digital Sciences Center, a Singapore-based research center affiliated with the

University of Illinois at Urbana-Champaign. His research interests include data driven networking, software defined networking (SDN), Internet measurement and monitoring, Peer-to-Peer content distribution, cloud computing and real-time distributed stream analytics.